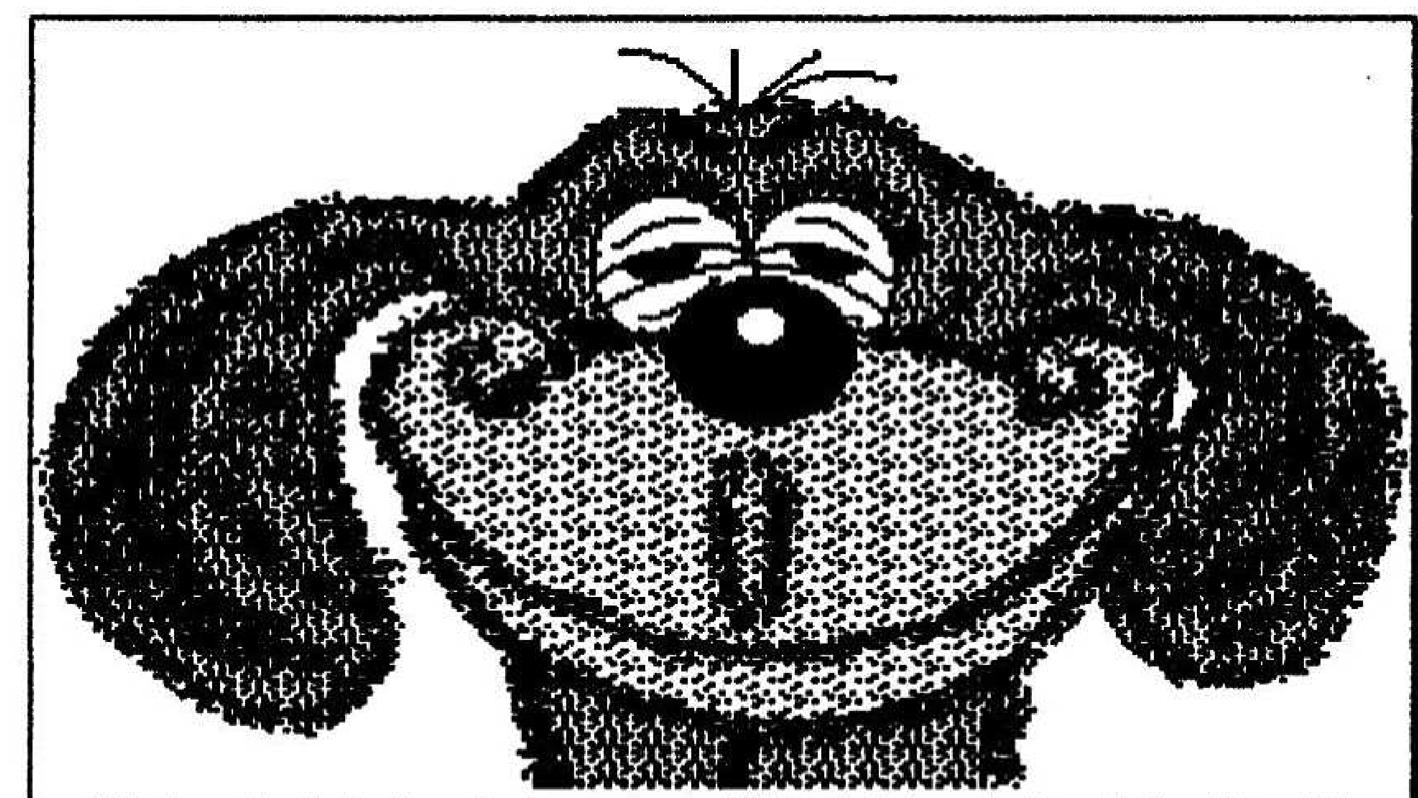
SPECTRUM PROFI CLUB

für Spectrum und SAM-User



"Ach wie ich das Leben mag ! Heut ist wieder Info Tag !"

Inhalt:

Wolfgang Haller
Ernastr. 33
5000 Köln 80
Tel. 0221/685946

INFO

2/91

Smalltalk

Mitglieder

Durch einen "Nachzügler" und drei Neueintritte erhöht sich die Mitgliederzahl auf 101 User (Hurra, die magische Grenze ist wieder gepackt!).

Hier die Anschriften:

Peter Lasker, Clematisstraat 50
Nl-7741 SM Coevorden

Friedrch D. Mehedinti, Burgstraße 51,
6750 Kaiserslautern

Dierk Reuter, Hermann-Kauffmann-Str.13
2000 Hamburg 60

Walter Sperl, Uferstraße 308,
A-2625 Schwarzau/Stfd.

Herzlich willkommen im SPC!

Thema Hobbit

Zum russischen Spectrum-Clone gibt es zur Zeit nichts Neues zu sagen. Wenn wir aber Neues erfahren sollten, teilen wir es euch mit.

Clubtreffen

Immer wieder werden wir auf das Thema Clubtreffen angesprochen. Auch wir finden, daß es wieder einmal an der Zeit wäre,ein Treffen zu organisieren. Zur Zeit läßt sich noch nicht genau sagen, wann es stattfinden kann. Geplant ist aber eins im Herbst nach der Ferienzeit. Ein Problem stellt zur Zeit auch noch die Räumlichkeit dar. Wir brauchen zumindest mehrere Stromanschlüsse. Möglicherweise legen wir das Clubtreffen wie damals wieder auf einen Computer-Flohmarkttag im alten Wartesaal des Kölner Hauptbahnhof, der auch verkehrsmäßig am günstigsten zu erreichen ist.
Was haltet ihr davon? Wer wäre dabei? Eventuell müßtet ihr aber einen kleinen finanziellen Beitrag dazu leisten.

Pokes

Crosswize = 51617,0 (Leben)
Shark = 49933,0 (unendlich Schild)
Ghostbusters II:
1. Level 49494,0 (unendlich Laser)
49624,0 (Laserschranke)
49172,0 (unendl.Schutzschild)
51754,0 (Aliens sterben)
2. Level 55877,0 (unendlich Schleim)
3. Level 53878,0 (unendlich Zeit)

Dirk Mayer, Köln

Softwareprojekt

Ich habe vor, das Programm Kings Quest 1 auf dem Spectrum zu realisieren. Die Szenen sollen von der Opus nachgeladen werden. Es werden Leute mit Zeit, Opus und Assemblererfahrung gesucht. Vor allem die Grafiksteuerung möchte ich vollständig abgeben. Bei Interesse bitte melden bei: D.Hucke, Korbacher Str.241, 3500 Kassel (nur schriftlich)

Projekt Opus/Beta

Der Fehler in der Umrechnung der Sektoren bei der Opus ist lokalisiert. Schuld an dem Rechenfehler ist der MC-Sektor der Opus. Man kann im Kanaleintrag der Opus die "geheimen Sektoren" angeben. Normal eins. Wird dieser Wert auf null gesetzt, wird die Track-/Sektorberechnung korrekt durchgeführt.

Dieter Hucke

Proj. Opus/Disciple

Danke an Rudolf Pirsch für die Zusendung der Disciple-Disk. Lesen kann ich sie bereits, ich brauche aber Angaben über den Katalogtrack der Disciple. Kann ein Clubmitglied hier im Info mal eine Serie darüber bringen? D.Hucke

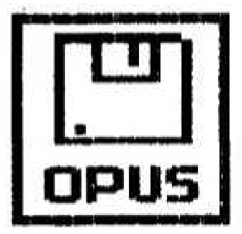
P.S.: Grüße nach Ostdeutschland! Ich hoffe, daß ihr mittlerweile doch endlich das Info 1/91 erhalten habt, der Poststreik ist ja jetzt vorbei. WH.

Ich habe viele Disketten, die häufig benutzt werden und es ist nicht ungewöhnlich, daß ich 60 oder 70 Files darauf habe. Natürlich sind die Files in keiner logischen Reihenfolge, und so ist es manchmal nicht leicht, etwas zu finden. "Reorg" ist ein kleines, in Basic geschriebenes Programm, welches einen Ausdruck des Directories auf einem Drucker erlaubt. Doch nicht nur das. "Reorg" läßt dem User die Wahl, welches Programm als erstes, als zweites usw. auf der Diskette gezeigt werden sollte. Alle Programme, die nicht gewählt werden, werden einfach an das Ende des Directories angehängt.

Ich hoffe, daß dieses Programm euch hilft, ein bißchen Ordnung in eure Diskettensammlung zu bringen.

```
* REORG *
  10 REM  SAM COUPE DIRECTORY REORG
  20 REM  (c) 1990 Spencer
  30 REM
1000 DIM D(80),F(80)
1010 head
1020 clarr
1030 insdk
1040 PRINT #0;  AT  1,0;"Directory  to
     Printer (y or n) ?": GET a$
1050 IF a$="y" OR a$="Y" THEN OPEN #2;
     "p"
1060 DIR 1
1070 OPEN #2;"s"
1080 FOR a=1 TO 80
1090 PRINT AT 17,0; INVERSE 1;"      Di
     rectrory Position ";a;" ?   "
1100 INPUT "File Nr. (0=end,$=restart)
      ?"; LINE a$
1110 IF a$="$" THEN GOTO 1000
1120 IF a$(1)<"0" OR a$(1)>"9" THEN BE
     EP .1,5: GOTO 1100
1130 LET file=VAL (a$)
1140 IF file=0 THEN GOTO 1190
1150 IF file<0 OR file>80 THEN BEEP .1
     ,5: GOTO 1100
1160 LET D(a)=file
1170 LET F(file)=0
1180 NEXT a
1190 PRINT AT 17,0; INVERSE 1;"
       READING HEADERS            "
1200 FOR b=1 TO 80
1210   IF F(b)<>0
1220    LET D(a)=F(b),a=a+1
1230   END IF
1240 NEXT b
1250 readhead
1260 PRINT AT 17,0; INVERSE 1;"
       REORGANIZING               "
1270 FOR a=1 TO 80
1280   LET file=D(a)
1290    IF a=(INT (a/2)*2)
1300     POKE 50256,MEM$(99744+(file*2
         56) TO 99743+((file+1)*256))
1310     WRITE AT 1,(INT ((a-1)/20)),(a/
          2-1) MOD 10+1,50000
1320       GOTO 1360
1330     ELSE
1340      POKE 50000,MEM$(99744+(file*2
          56) TO 99743+((file+1)*256))
1350     END IF
1360 NEXT a
1370 PRINT AT 17,0; INVERSE 1;"
       REORG. FINISHED            "
1380 PAUSE 200
1390 DIR 1
1400 STOP
5000 DEF PROC head
5010 CLS
5020 PRINT AT 0,7;" DISK REORGANIZER";
     AT 1,8;"(c) 1990 Spencer"
5030 END PROC
5100 DEF PROC clarr
5110 FOR a=1 TO 80
5120 LT F(a)=a
5130 NEXT a
5140 END PROC
5200 DEF PROC insdk
5210 PRINT AT 17,0; Inverse 1;"
     INSERT SAM DISKETTE        ";AT 18,0;
     "       Press any key          "
5220 BEEP .1,10
5230 PAUSE
5240 PRINT AT 17,0;"
                                "
5250 END PROC
5300 DEF PROC readhead
5310 LET adr1=100000
5320  FOR track=0 TO 3
5330   FOR sect=1 TO 10
5340    READ AT 1,track,sect,50000
5350    POKE adr1,MEM$(50000 TO 50511)
5360    LET adr1=adr1+512
5370   NEXT sect
5380  NEXT track
5390 END PROC
9999 CLEAR
     SAVE OVER "REORG" LINE 10
```

Ian D. Spencer, Fichtenweg 10c, 5203 Much, Tel. 02245/1657



Die Opus Discovery
Teil 8

Nach einem Monat Pause wollen wir uns heute damit beschäftigen wie man mit der Opus in Assembler LOADen und SAVEn kann. Dazu benötigen wir aus den Sprungtabellen die Funktionen OPENCH, CLOSCH und SAVECH, die ich zuerst einmal erklären möchte bevor ich ein Beispielprogramm in Assembler vorstellen möchte.

Dabei möchte ich mich auch ganz herzlich bei Dieter Hucke bedanken, der mir sehr bei dem Problem geholfen hat, ein funktionsfähiges Assemblerprogramm zu schreiben.

So, nun aber zu den drei Funktionen:

OPENCH:

Mit dieser Funktion werden auf der Opus Schreib- und Lesekanäle geöffnet. Die Öffnung eines solchen Kanales ist erst einmal die Voraussetzung, daß die Opus Programme von der Diskette laden oder speichern kann. Dabei müssen jedoch erst einmal einige Daten in bestimmte Adressen schreiben.
Als erstes wäre dort ein Parameterblock, der genauso aussieht wie der bei der CALUTL-Routine (RU 12/90). Der Dateiname ist dabei der Name des Programms, das wir speichern oder laden wollen. Mit Hilfe von RST 48 im Speccy-Rom schaffen wir uns dann einen Platz im Workspace. Dabei steht in BC die Länge unseres Parameterblocks und als Ergebnis der Routine erhalten wir die Anfangsadresse unseres Workspaces. In HL kommt dann die Anfangsadresse von unserem Parameterblock und danach wird der Block mit dem Blockbefehl LDIR in den Workspace kopiert (im letzten Teil meiner Serie war hier ein kleiner Fehler: ich habe dort geschrieben, daß DE mit der Anfangsadresse des Parameterblocks geladen wird). Dann müssen Adresse (DE) und Länge (BC) des Parameterblocks auf den Kalkulatorstapel gebracht werden (mit CALL 10934 (2AB6h) im Speccy-Rom). Dann kommt die Länge des Kanals (nicht identisch mit der Länge des Programms, das wir speichern wollen!) in BC und mit Hilfe von CALL 11563 (2D2Bh) auch auf diesen Stapel. Da uns diese Länge nicht bekannt ist laden wir ganz einfach eine 0 nach BC. Nachdem wir BC (also die 0) auf den Stapel gelegt haben, schreiben wir diese Länge noch nach HL. Als letztes müssen in A bestimmte Bits gesetzt werden, die folgende Bedeutung haben:

Bit 0: Ausgabe über den Kanal
Bit 1: Eingabe über den Kanal
Bit 2: Anlegen eines Kanals

Bit 0 hat für LOAD und SAVE keine Bedeutung. Bits 3 bis 7 sollten zurückgesetzt sein. Das bedeutet für LOAD, MERGE und VERIFY: LD A,2; für SAVE: LD A,4. Dann endlich kann OPENCH aufgerufen werden. Als Ergebnis erhalten wir im IX-Register die Position des Kanals. Da wir diese Position für den Aufruf von SAVECH und CLOSCH benötigen PUSHen wir IX.

SAVECH:

Auch für SAVECH müssen einige Dinge beachtet werden bevor man die Routine aufruft:

Zuerst einmal wird ein Header, der mit einem Header auf Kassette identisch ist in den Workspace kopiert.(wie dieser Header aufgebaut ist findet Ihr in RU 8/90) Die Kopie in den Workspace erfolgt genauso wie oben erklärt. Dabei muß man aber beachten, daß die Bytes 2-11 (=Name der Programms) leer bleiben. Außerdem müssen bei LOAD, MERGE und VERIFY 34 Bytes in den Workspace. Die ersten 17 Bytes sind dabei unser Header. Die restlichen 17 Bytes bleiben leer (dorthin kopiert die SAVECH-Routine später den Header, der auf Diskette gefunden wurde; dann vergleicht diese Routine beide Header). Nach HL laden wir anschliessend die Anfangsadresse unserer Daten (z.B. bei einem Screen: LD HL,16384). Jetzt besorgen wir uns mit POP IX das IX-Register wieder (IX sollte anschliessend direkt wieder gePUSHt werden, da wir es für CLOSCH nocheinmal brauchen). Als Letztes kommt in das A-Register ein Kommandobyte (0=SAVE, 1=LOAD, 2=VERIFY, 3=MERGE). Jetzt kann die SAVECH-Routine aufgerufen werden.

CLOSCH:

Als letztes muß unser Kanal wieder geschlossen werden. Das ist ganz einfach: Mit POP IX besorgen wir uns wieder den Kanalanfang. Danach kann schon CLOSCH aufgerufen werden.

Um jetzt alle Klarheiten zu beseitigen, ist hier noch ein Beispielprogramm, das einen SCREEN$ unter dem Namen "screen" von der Diskette lädt (dieses Programm einfach an Programm 1 aus RU 12/90 anhängen).

Programm 3:

```
RST 16
DEFW 5808 ; löscht den Workspace
LD BC,pare-para ; Länge P-Block
RST 16
DEFW 48 ; Raum im Workspace
PUSH DE
PUSH BC
LD HL,para
LDIR ; Block in Workspace
POP BC ; Länge Parameterblock
POP DE ; Adresse Parameterblock
LD A,0 ; ohne Bedeutung
RST 16
DEFW 10934 ; A, DE, BC auf Stapel
LD BC,0 ; Kanallänge unbekannt
PUSH BC
RST  16
DEFW 11563 ; BC auch auf Stapel
POP HL ; BC nach HL
LD A,2 ; = Kanaleingabe
CALL OPENCH

(weiter im Programm rechts oben ->)
```

```
       RST 16
       DEFW 5808 ; Workspace löschen
       LD BC,34 ; Headerlänge
       RST 16
       DEFW 48
       PUSH DE ; Anf.adr. Workspace
       LD HL,header
       LDIR ; Daten in Workspace
       POP DE
       LD HL,16384 ; Beginn der Daten
       LD A,1 ; 1=LOAD
       PUSH IX
       CALL SAVECH
       POP IX
       CALL CLOSCH
       LD SP,(23613) ; der Stackpointer
       DEC SP ; wird korrigiert
       DEC SP
       JP 5960 ; Opus aus und RET
para   DEFB "m"
       DEFB 1
       DEFM "screen    "
pare   NOP
header DEFB 3
       DEFS 10 ; 10 freie Bytes
       DEFW 0 ; 0=Länge unbekannt
       DEFW 0 ; 0=Startadresse unbek.
```

So, das war es mal wieder für heute, bis bald

Rüdiger Döring, Meisenstraße 10, 5467 Vettelschoß, Tel. 02645/3060

Spectrum 128 und Musik (Teil 1)

Hallo Musik-Freunde!

Die ständigen Anfragen nach 128er-Musikstücken (u.a von Rüdiger Döring in RU 11/90) haben mich dazu inspiriert, einmal etwas zu dieser Problematik zu schreiben. Selbstverständlich gibt es noch mehr Leute, die den wertvollen Soundchip auch benutzen. Dieser Beitrag ist für alle gedacht, die selber einmal Musik auf dem 128er programmieren wollen. Für die 48er-User ist es sicher interessant zu erfahren, daß der 128er außer seinem größeren Speicher noch andere Vorteile hat. Vielleicht könnte man auf diesem Wege zu einem Musik-Programmierwettstreit aufrufen?

Zur vollwertigen Musikerzeugung waren die 128er-Vorgänger eigentlich kaum geeignet. Der ZX 81 war von Natur aus stumm und der 48er-Standart-Spectrum war mehr zum Morsen geeignet als zum Musizieren. Trotzdem gab und gibt es findige Programmierer, die mttels (softwaremäßiger) Modulationen erstaunliche Soundeffekte produzieren, selbst vor Sampling (Einlesen natürlicher Klänge oder Geräusche über die EAR-Buchse und deren Abspeichernung und Wiedergabe) wird da nicht zurückgeschreckt.

Mit dem 128er hat man nun die akustische Schwachstelle beseitigt und einen Soundchip vom Typ AY-3-8912-A von General Instruments (im folgenden PSG "Programmierbarer Soundgenerator" genannt) eingebaut. Und nicht nur das, man hat auch gleich unter Einsparung der User-Grafikzeichen "T" und "U" den BASIC-Befehlsvorrat entsprechend erweitert und das leistungsfähige Kommando "PLAY" implementiert, so daß man in Basic komplette Musikstücke schreiben und dabei alle Funktionen des PSG programmieren kann (das ist bei vielen anderen BASIC-Computern z.B. C-64 nur mit hunderten von POKEs möglich).

Im folgenden möchte ich nun auf die Musikprogrammierung des PSG im 128er-BASIC, die MIDI-Möglichkeiten des 128er, die einzelnen Register des PSG und die direkte Programmierung des PSG in MC eingehen.

Zunächst aber zur BASIC-Programmierung: Der PSG kann Töne und Rauschen auf 3 unabhängigen Kanälen erzeugen. Dabei kann für jeden Kanal die Lautstärke getrennt programmiert werden. Weiterhin ist ein einfacher Hüllkurvengenerator vorhanden, mit dem man den Lautstärkeverlauf von einem oder mehreren Kanälen steuern kann (An- und Abschwellen des Tons oder Geräusches). Die Hüllkurvenform und die An- bzw. Abschwellzeit ist ebenfalls programierbar.

Das Erzeugen von Musik im 128er ist im Prinzip ganz einfach. Voraussetzung ist allerdings eine gewisse Notenkenntnis. Man schreibt dann die zu spielenden Noten in einen String und "spielt" den String mit dem Befehl PLAY ab:

z.B. `LET a$="cde": LET b$="efg": LET c$="O3cde": PLAY a$,b$,c$`

Bei kurzen Strings kann man auch vereinfachen:

z.B. `PLAY "cde","efg","O3cde"`

Das Prinzip ist immer das gleiche, nur der Inhalt der Strings wird logischerweise unterschiedlich sein. Es müssen nicht unbedingt alle 3 Kanäle verwendet werden, der PLAY-Befehl spielt auch 1 oder 2 Stimmen. Im PLAY-Befehl können sogar bis zu 8 Strings enthalten sein, aber dazu später mehr Informationen.

Nun zum Inhalt der Strings: Es existieren eine Reihe von Subkommandos in Form von Buchstaben und Zahlen, die später vom PLAY-Befehl für den PSG entsprechend aufbereitet werden. Die wichtigsten sind dabei die Buchstaben a-g und A-G sowie die Zahlen 1-12. Damit lassen sich bereits alle Noten programmieren. Die Buchstaben haben im einzelnen folgende Bedeutung:

c,d,e,f,g,a,b --> entsprechen der Tonleiter c,d,e,f,g,a,h
C,D,E,F,G,A,B --> entsprechen der Tonleiter der darüberliegenden Oktave (auf dem Keyboard rechts daneben)

Beachte die Umschreibung b für den Ton h bzw. B für H!

Werden nur diese Buchstaben im String verwendet, so spielt der PLAY-Befehl Viertel-Noten im Bereich von c1 bis h2. Das ist genau der Bereich, der auf der Melodie-Notenzeile dargestellt wird und normalerweise für die Melodiestimme(n) benutzt wird.

Nur Viertelnoten klingen etwas langweilig, deshalb nehme ich nun die oben erwähnten Zahlen zu Hilfe. Diese bestimmen nämlich die Tonlänge:

1-> Sechzehntel	5-> Viertel	9-> Ganze
2-> verlängerte Sechzehntel	6-> verlängerte Viertel	10-> Sechzehntel-Triole
3-> Achtel	7-> Halbe	11-> Achtel-Triole
4-> verlängerte Achtel	8-> verlängerte Halbe	12-> Viertel-Triole

Kurze Erklärung für die Nichtmusikalischen: Die Zahlen bestimmen die Zeitdauer. Es gilt: 2 Halbe = 1 Ganze, 2 Viertel = 1 Halbe, usw., die verlängerten Noten (auf dem Notenblatt mit Punkt dargestellt) haben die 1.5-fache Zeitdauer, z.B. 1 Viertel + 1 Achtel = 1 verlängerte Viertel = 3 Achtel (simple Bruchrechnung). Bei einer Triole werden 3 verkürzte Noten in der gleichen Zeit gespielt, die sonst für 2 Noten erforderlich ist, z.B. werden bei 1 Achtel-Triole 3 statt 2 Achtelnoten in der gleichen Zeit gespielt.

Um nun eine bestimmte Note zu spielen, wird vor den Notenbuchstaben einfach die entsprechende Zahl geschrieben (z.B. spielt "3d" eine Achtelnote d). Diese Zahl gilt dann für alle folgenden Noten bis durch eine andere Zahl ein neuer Wert eingestellt wird. Bei Triolen (10-12) gilt der Wert nur für die nächsten 3 Noten, danach ist die vorherige Einstellung wieder wirksam. Nach einer Notenlängen-Zahl muß aber in jedem Fall ein Notenbuchstabe folgen.

Es ist auch möglich, andere Zeiten als die mit den Zahlen 1-9 vorgegebenen einzustellen. Dazu können die Zahlen mit Hilfe des Unterstriches "_" verbunden werden, z.B. eine Notenlänge von 5 Sechzehntel ist eine Kopplung von 1 Viertel und 1 Sechzehntel, also wird 5_1 vor den Notenbuchstaben geschrieben. Der zweite Wert ist dann die Grundeinstellung für die folgenden Noten.

Ein weiteres wichtiges Zeichen ist das Und-Symbol "&". Dieses Zeichen kann anstelle eines Notenbuchstaben stehen und ersetzt die Note durch eine gleichlange Pause. Während dieser Zeit wird kein Ton erzeugt. Die Zeitdauer kann wie bei einer Note durch eine vorangestellte Zahl (1-12) bestimmt werden.

Außer den ganzen Tönen der C-Dur-Tonleiter gibt es auch noch Halbtöne (für Nichtmusikalische: die schwarzen Tasten auf dem Klavier). Um solche Noten zu spielen muß, abhängig davon, ob der darüber- oder darunterliegende Halbton gespielt werden soll, vor dem Notenbuchstaben ein Doppelkreuz "#" oder ein Dollarzeichen "$" stehen. Wird z.B. #g geschrieben, so wird ein gis (Halbton über g) gespielt, schreibt man $g, so ertönt ein ges (Halbton unter g).

Wie oben beschrieben, lassen sich die Notenbuchstaben ideal zur Programmierung einer Melodiestimme verwenden. Wie kann man nun eine Baßstimme oder noch höhere Notenwerte erzeugen? Dazu gibt es den Kommandobuchstaben "O". Damit läßt sich die Oktave, in der die Notenbuchstaben erklingen, verschieben. Nach dem "O" muß dann die Oktavnummer im Bereich von 0 bis 8 angegeben werden. Es gilt folgende Oktav-Zuordnung:

Oktavnummer	Notenbereich c-b	Notenbereich C-B	Frequenz (Hz)
O0	C3 - H3	C2 - H2	8.176 - 30.87
O1	C2 - H2	C1 - H1	16.35 - 61.74
O2	C1 - H1	C - H	32.70 - 123.5
O3	C - H	c - h	65.41 - 246.9
O4	c - h	c1 - h1	130.8 - 493.9
O5	c1 - h1	c2 - h2	261.6 - 987.8
O6	c2 - h2	c3 - h3	523.3 - 1975.5
O7	c3 - h3	c4 - h4	1046.5 - 3951.1
O8	c4 - h4	c5 - h5	2093.0 - 7902.1

Das muß fürs Erste genügen. Im nächsten Teil geht es dann um weitere String-Kommandos. Tschüß bis zum nächsten mal

Scott-Falk Hühn, Erich-Heyl-Str. 4, Sömmerda / Thüringen, O-5230
Tel.(O): 00626 22467 / Tel.(W): 0037 626 22467

Liebe Specci-Freunde!

Der Befehlssatz des Zilog Z 80 / Teil 7

LD r,(IX + d)

Lade das Register r indirekt aus der indiziert adressierten Speicherzelle (IX + d).
Der Inhalt der Speicherzelle, die durch das Register IX plus dem gegebenen Offset adressiert wird, wird in das Register r geladen.
Beispiel: *LD E,(IX + 5)*

LD r,(IY + d)

Lade das Register r indirekt aus der indiziert adressierten Speicherzelle (IY + d).
Der Inhalt der Speicherzelle, die durch das Register IY plus dem gegebenen Offset adressiert wird, wird in das Register r geladen.
Beispiel: *LD A,(IY + 2)*

LD (IX + d),n

Lade die indiziert adressierte Speicherzelle (IX + d) mit den unmittelbaren Daten n.
Der Inhalt der Speicherzelle, die unmittelbar auf den Op-Code folgt, wird in die Speicherzelle geladen, die durch das Register IX plus dem gegebenen Offset adressiert wird.
Beispiel: *LD (IX + 4),FF*

LD (IY + d),n

Lade die indiziert adressierte Speicherzelle (IY + d) mit den unmittelbaren Daten n.
Der Inhalt der Speicherzelle, die unmittelbar auf den Op-Code folgt, wird in die Speicherzelle geladen, die durch das Register IY plus dem gegebenen Offset adressiert wird.
Beispiel: *LD (IY + 3),BA*

LD (IX + d),r

Lade die indiziert adressierte Speicherzelle (IX + d) aus dem Register r.
Der Inhalt des angegebenen Registers wird in die Speicherzelle geladen, die durch das Register IX plus dem gegebenen Offset adressiert wird.
Beispiel: *LD (IX + 1),C*

LD (IY + d),r

Lade die indiziert adressierte Speicherzelle (IY + d) aus dem Register r.
Der Inhalt des angegebenen Registers wird in die Speicherzelle geladen, die durch das Register IY plus dem gegebenen Offset adressiert wird.
Beispiel: *LD (IY + 3),A*

LD A,(nn)

Lade den Akkumulator aus der Speicherstelle nn.
Der Inhalt der Speicherzelle, die durch den Inhalt der beiden Bytes nach dem Op-Code adressiert wird, wird in den Akkumulator geladen. Das untere Byte der Adresse folgt unmittelbar auf den Op-Code.
Beispiel: *LD A,(3301)*

LD (nn),A

Lade die direkt adressierte Speicherstelle (nn) aus dem Akkumulator.
Der Inhalt der Speicherzelle, die durch den Inhalt der beiden Bytes nach dem Op-Code adressiert wird, wird aus dem Akkumulator geladen. Das untere Byte der Adresse folgt unmittelbar auf den Op-Code.
Beispiel: *LD (0321),A*

LD (nn),dd

Lade die durch nn adressierten Spicherstellen aus dem Registerpaar dd.
Die untere Hälfte des angegebenen Registerpaares wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den

Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-code.
Beispiel: *LD (040B),BC*

LD (nn),HL
Lade die durch nn adressierten Speicherstellen aus dem Register HL.
Der Inhalt des Registers L wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Das Register H wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD (40B9),HL*

LD (nn),IX
Lade die durch nn adressierten Speicherstellen aus dem Register IX.
Die untere Hälfte des Register IX wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD (012B),IX*

LD (nn),IY
Lade die durch nn adressierten Speicherstellen aus dem Register IY.
Die untere Hälfte des Register IY wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD (BD04),IY*

LD A,(BC)
Lade den Akkumulator aus der durch das Registerpaar BC indirekt adressierten Speicherstelle.
Der Inhalt der Speicherzelle, die durch den Inhalt des Registerpaares BC adressiert wird, wird in den Akkumulator geladen.
Beispiel: *LD A,(BC)*

LD A,(DE)
Lade den Akkumulator aus der durch das Registerpaar DE indirekt adressierten Speicherstelle.
Der Inhalt der Speicherzelle, die durch den Inhalt des Registerpaares DE adressiert wird, wird in den Akkumulator geladen.
Beispiel: *LD A,(DE)*

LD A,I
Lade den Akkumulator aus dem Interruptvektor-Register I.
Der Inhalt des Interruptvektor-Registers I wird in den Akkumulator geladen.
Beispiel: *LD A,I*

LD I,A
Lade das Interrupt-Register I aus dem Akkumulator.
Der Inhalt des Akkumulators wird in das Interruptvektor-Register I geladen.
Beispiel: *LD I,A*

LD A,R
Lade den Akkumulator aus dem Memory-Refresh-Register R.
Der Inhalt des Memory-Refresh-Registers wird in den Akkumulator geladen.
Beispiel: *LD A,R*

LD HL,(nn)
Lade das Register HL aus der Spicherzelle nn.
Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird ins Register L geladen. Das Register H wird aus der Speicherstelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD HL,(0024)*

LD IX,nn
Lade das Register IX mit den unmittelbaren Daten nn.
Der Inhalt der Speicherzellen, die unmittelbar auf den Op-Code folgen, wird in das Register IX geladen. Das untere Byte erscheint unmittelbr hinter dem Op-Code.
Beispiel: *LD IX,B0B1*

LD IX,(nn)
Lade das Register IX aus der Speicherzelle nn.
Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird in die untere Hälfte des Registers IX geladen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD IX,(010B)*

LD IY,nn
Lade das Register IY mit den unmittelbaren Daten nn.
Der Inhalt der Speicherzellen, die unmittelbar auf den Op-Code folgen, wird in das Register IY geladen. Das untere Byte erscheint unmittelbr hinter dem Op-Code.
Beispiel: *LD IY,21*

LD IY,(nn)
Lade das Register IY aus der Speicherzelle nn.
Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird in die untere Hälfte des Registers IY geladen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.
Beispiel: *LD IY,(500D)*

Das war's. Bis zur Ausgabe Nr. 8 viel Spaß

Harald R. Lack, Heidenauer Str. 5, 8201 Raubling

Pascal auf dem Spectrum

Einführungskurs by Lord Luxor

Zuerst einmal die Story, wie ich zu PASCAL auf dem Speccy kam. Meine ersten Schritte in PASCAL tat ich auf einem XT mit TurboPascal. Schon bald war ich begeistert von PASCAL, wegen der Vielfalt der Befehle und der Einfachheit. Dummerweise besitze ich keinen PC und auf meinem Amiga (den ich damals eh noch nicht hatte) gibt es kein gescheites PASCAL. So kam ich an Hisoft Pascal. Mit diesem arbeite ich auch sehr gerne, auch wenn es einige Nachteile hat.
In diesem Kurzeinweisungs-Kurs beziehe ich mich stets auf Hisoft Pascal 4TM1.6, ich kenne nicht genau die Unterschiede, man (und Frau) möge mir das nachsehen.
Ich möchte auch gleich die Chance nutzen und fragen, ob sich jemand die Mühe gemacht hat und dieses Pascal auf Beta-Disk umgeschrieben hat. Daran wäre ich doch sehr interessiert. Ich gehe mal davon aus, daß einige User auf irgendeiner staubigen Kassette, Diskette oder Cartridge Pascal besitzen. Ich erspare mir die Ladezeremonie, das müßt ihr selbst rausfinden.
Sofern es keinen Ladefehler gab, erscheint auf dem Bildschirm die Frage "Top of RAM?" - nicht beachten, ein freudig, kraftvolles ENTER, genauso wie bei "Top of RAM for T?" und "Table Size?". Nun erscheint "Copyright Hisoft 1983,84 all rights reserved" (was schon so alt?!) und ein Prompt ">" mit einem blinkenden C-Cursor. Was nun? Versuchen wir es einmal mit PRINT "HALLO". Wie ein wohlerzogener Franzose sagt unser Speccy "Pardon?", was auf Umgangsdeutsch "Häh?" heißt.
Was war falsch? Zwei Dinge: 1. gibt es in Pascal nicht den PRINT-Befehl und 2. befinden wir uns im sogenannten Befehlsmodus, der keine Pascal-Befehle kennt, also selbst wenn PRINT ein Pascal-Befehl wäre hätte unser Speccy ihn nicht verstanden.

Wie können wir Jetzt aber Pascal-Befehle eingeben? Dazu müssen wir erstmal in den sogenannten Editor. Dies geschieht mit I -ENTER- (I für Insert). Machen wir das mal (ich rede immer "wir", weil ich vor dem Speccy sitze und alles mitmache). Nun steht da eine 10 und dahinter der Cursor. Das ist der Editor und wir befinden uns in Zeile 10. Halt, bevor Jemand beginnt wild Befehle einzugeben, der Editor überprüft den Syntax der Befehle nicht.
Beginnen wir mit einem Super-Programm, das Hallo auf den Bildschirm schreiben kann.
Bei Pascal muß in der ersten Zeile immer der Programmname definiert werden. Das geschieht mit PROGRAM -Name-;. Wichtig ist das ;, damit bekommt der Compiler später das Zeichen Ende dieses Befehls. Wir beenden die Zeile mit ENTER.
Wir haben schon den wichtigsten Befehl gelernt, denn ohne den läuft nichts, bzw. beim Compilieren gibt es eine Latte von Fehlermeldungen.
Bevor das eigentliche Programm beginnt muß man den Compiler darauf aufmerksam machen, mit dem Befehl BEGIN;. Dieser zeigt dem Compiler den Beginn einer Prozedur, Funktion, Schleife oder ähnlichem. ENTER!
In BASIC lernt man zuerst meist den PRINT-Befehl kennen, in Pascal müßte es Ja auch so etwas geben, klar, es heißt hier einfach WRITELN('-Text-'); und unterscheidet sich etwas vom PRINT-Befehl, aber davon später. ENTER!
Beenden wir dieses Programm mit END. Wichtig ist hier der Punkt. Der signalisiert das Ende des Programms, denn END kann auch für das Ende einer Prozedur etc. stehen.
Verlassen wir nun den Editor mit CAPSSHIFT - 1.
Jetzt haben wir die Wahl, ob wir das Programm abspeichern oder compilieren lassen wollen. Alte Speccyuser-Regel: Erst saven, dann pfuschen!
Der Befehl (wir sind wieder im Befehlsmodus, also kein Pascal-Befehl) lautet: P Anfangszeile,Endzeile,Name
Bei unserem Programm würde das so aussehen: P 10,40,Hallo.
Eine kleine Warnung für 4TM1.6 user, nach dem Befehl nicht gleich ENTER drücken, da diese Pascal-Version nicht wartet, sondern den Befehl sofort ausführt. Beim Saven werden einige feststellen, daß Pascal ein eigenes Kassettenformat benutzt. Ist alles fehlerlos abgesavt compilieren wir unser erstes Programm mit dem Befehl C Anfangszeile, gibt man keine Anfangszeile an, so compiliert unser Speccy ab der ersten Zeile, also geben wir nur C ein. Es erscheint unser Programm, links davon einige merkwürdige Zeichen, Assemblerfreks werden sie vertraut vorkommen. Unter unserem Listing erscheint die Meldung "End Adress AD87" (Jedenfalls bei mir), damit ist klar, daß diese Zeichen die Adressen sind, in denen der Jeweilige Befehl compiliert abgelegt wurde.
Nun fragt unser Speccy "Run?", was wir mit Y beantworten und wenn ihr keinen Fehler gemacht habt, seht ihr wie ich ein Hallo auf dem Bildschirm, toll was? Unser erstes Programm hat funktioniert! Na, das war Jetzt alles sehr einfach und primitiv, im nächsten Heft wird alles komplizierter.

Eine kleine Zusammenfassung dessen, was wir heute gelernt haben:

<u>Befehle des Befehlsmodus:</u>

I a,b	springt in den Editor. a=Anfangszeile, b=Schrittgröße der Zeilennr.
P a,b,name	speichert Quellcode (Programmtext) von Zeile a bis Zeile b unter dem Name name aufs Band
C a	Compiliert Quellcode ab Zeile a

<u>Pascal-Befehle:</u>

PROGRAM -name-;	Definiert Name des Programms im Editor, immer in der 1. Zeile vereinbaren!
BEGIN;	Definiert Blockbeginn einer Prozedur, Funktion oder sonstigen Programmteilen
WRITELN('-Text-');	wie in BASIC PRINT
END(;)	Ende einer Prozedur, Funktion oder Schleife
END.	Ende des Programms, immer letzte Zeile!

So, im nächsten Teil weniger Text, mehr Befehle. Ich hoffe, einige User werden ihr Pascal entstauben. Bei Fragen könnt ihr euch an mich wenden:

LORD LUXOR, PLK 185207 E, 6900 Heidelberg

Sortierprogramm

Hallo Freaks.
Heute ein kleines Sortierprogrämmchen. Häufig kommt es vor, daß man eine Liste von irgendwas erstellen möchte. Zum Bleistift eine Namensliste. Will man diese schön geordnet haben, muß man vor der Erstellung das Ganze vorsortieren.
Folgendes Programm erspart dieses vorsortieren. Das Programm zeichnet sich durch das Fehlen jeglichen Komforts aus. Es unterscheidet weder Groß-/Kleinschreibung noch Zahlen und es sortiert nur nach dem ersten Buchstaben. Also genau das richtige für einfache Listen. Nach Start des Programmes bei der Frage Name ? das gewünschte eingeben und Enter. Zum sortieren STOP (Symbolshift + A) drücken und Enter. Keine Angst, wenn nun im Screen was passiert. Der Screen wird als Zwischenspeicher benutzt. Da das Sortieren in Maschinensprache ausgeführt wird, dauert es nicht lange. Anschließend kann man sich die sortierte Liste auf dem Screen anschauen oder abspeichern. Abgespeichert wird das Ganze als CODE-File, welchen man in's Tasword 2 oder Lastword laden kann. Hier muß der File jedoch noch "verschönert" werden. Dies bleibt aber jedem selbst überlassen.
Noch ein Hinweis: da das * als Eintragsendemerker für das MC-Programm dient, darf es nicht Bestandteil eines Eintrages sein!
Auch darf die Summe der Einträge nicht größer als 6912 Zeichen sein. Hier erinnert das Programm aber bei Überschreitung den Benutzer daran.

Hier das Programm "Sort":

P

```
   1 CLEAR 31999
   2 LET poke=32000
   3 CLS: PRINT "MOMENT BITTE": GO SUB 8000: CLS
  10 PRINT "     SORTIERPROGRAMM              Dieses Programm sortiert Daten  (kei
     ne Zahlen) und erzeugt einenTasword II File."
  11 PRINT "Es sortiert nur den 1.! Buchsta=ben. Die Eingabe kann in Klein- oder
     Grossschreibung erfolgen."
  12 PRINT "Der 1. Buchstabe wird autom. um=gewandelt in gross."
  13 PRINT "Das Programm ist zum erstmaligenAnlegen einer Datei (z.B. Pro=  gram
     mliste)im Tasword Format ge=dacht."
  14 PRINT "Eine hiermit erstellte Liste    muss jedoch noch mit Tasword    "ver
     schoenert" werden."
  15 PRINT "Dazu laedt man die sortierte Li=ste ins Tasword, faehrt mit dem Curs
     or hinter die entsprechende Stelle und drueckt Symbolshift +Y"
  16 PRINT "Damit wird der Rest der Zeile   an den Beginn der naechsten Zei=le g
     eschaufelt. Zugegeben, noch nicht ganz der wahre Jakob, aberimmer noch bess
     er(und schneller)als das zu Fuss sortierte Einge=ben in Tasword!"
  20 PRINT #0;"PRESS KEY TO BEGIN": PAUSE 0: CLS
 100 REM schreiben
 105 LET a$=""
 110 INPUT "Name : ";a$
 111 IF a$=" STOP " THEN POKE poke,226: POKE poke+1,226: POKE poke+2,226: RANDOM
     IZE poke-32000: GO TO 500
 115 IF poke>=38912 THEN PRINT "  DATEI VOLL  ": STOP
 120 LET a$=a$+" * "
 121 LET b=CODE a$(1): IF b>96 THEN LET b=b-32: LET a$(1)=CHR$ b
 129 PRINT AT 20,0;a$: RANDOMIZE USR 3280
 130 FOR n=1 TO LEN a$: POKE poke,CODE a$(n): LET poke=poke+1: NEXT n
 140 GOTO 100
 500 CLS
 501 PRINT AT 10,0;"Ich sortiere. Dazu benutze ich   als Zwischenspeicher den SC
     REEN$ ": PAUSE 30
 510 RANDOMIZE USR 23300
 511 POKE 23658,0
 520 CLS: PRINT "              FERTIG                Willst du das sortierte
     L)esen      S)peichern ?"
 550 IF INKEY$="l" THEN GOTO 9100
 551 IF INKEY$="s" THEN GOTO 9000
 555 GO TO 550
```

```
8001 DATA 237,75,118,92,197,62,65,245,17,0,64,33,0,125,241,245,190,40,44,35,11,1
     20,177,40,16,62,226,190,40,11,62,42,190,32,240,35,35,11,11,24,229,241,193,1
     97,60,245,254,91,40,2,24,215,241,193,33,0,64,17,0,125
8002 DATA 237,176,201,126,54,32,18,35,19,11,62,42,190,32,244,18,35,19,11,126,18,
     35,19,11,126,254,226,40,208,24,179
8010 RESTORE 8000
8020 FOR n=23300 TO 23390: READ z: POKE n,z: NEXT n: RETURN
9000 REM saven
9001 LET la=poke-32000: PRINT "M)icrodrive      C)assette"
9002 IF INKEY$="m" THEN GO TO 9030
9003 IF INKEY$="c" THEN GO TO 9020
9005 GO TO 9002
9020 SAVE "p"CODE 32000,la: STOP
9030 SAVE *"m";1;"p"CODE 32000,la: STOP
9100 REM print
9110 FOR n=32000 TO poke
9120 PRINT CHR$ PEEK n;: NEXT n
9130 PRINT #0;"PRESS KEY": PAUSE 0: GO TO 520
9150 STOP
9200 PRINT AT 10,0: FOR n=16384 TO 18000: PRINT CHR$ PEEK n;: NEXT n
9300 STOP
9998 SAVE "sort" LINE 1
9999 SAVE *"m";1;"Tassort"
```

Also sortiert mal schön.....Paule Panther

Paul Webranitz, Borgasse 16, 5561 Kinheim, Tel. 06532/2607

Dieter Hucke, Korbacherstraße 241, 3500 Kassel

OPUS INTERN (1)

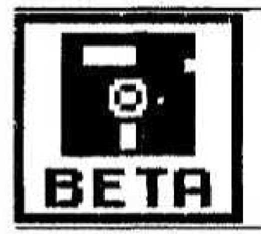
An dieser Stelle möchte ich mit einer Serie über das Innenleben der OPUS beginnen.
Wir werden bei der Stromversorgung anfangen, und bei den einzelnen Signalen auf dem Bus aufhören. Meistens wird es eine EINLEITUNG geben, die für den überblick reichen sollte. Der zweite Teil, EINGEMACHTES, geht an ebendasselbe, natürlich nur, wenn ich es selbst kapiert habe. Ich hab nämlich manches absolut noch nicht kapiert.
Manchmal wird Software auftauchen. Das ist logisch, denn die OPUS läuft nun mal mit Software. Aber genauere Beschreibungen bitte ich aus Rüdiger Dörings OPUS-Kurs zu nehmen, denn da steht sehr viel drin! So, genug der Vorrede...

<u>Die Stromversorgung :</u>

Einleitung :
Die OPUS hat ein eigenes Netzteil (Trafo, GleichRiechtEr, Spannungsstabilisatoren), mit dem sie die Laufwerke, den Spektrum und sich selbst mit Strom versorgt.
Es werden 12 V stabil, 5 Volt stabil und 9 V unstabil erzeugt. Die 9 Volt gehen an den Spektrum, der daraus seine eigenen Spannungen erzeugt, die 12 Volt an die Laufwerke, und die 5 Volt an die Laufwerke und die OPUS-Elektronik.

Eingemachtes :
Es gibt meines Wissens zwei grundlegende Versionen der Stabilisierung; einmal die Verwendung von Festspannungsreglern, 7812 und 7805, oder zweimal LM 317, mit Zenerdioden. Die Brückengleichrichter bzw Dioden sind geizig bemessen, und können überhitzen, ebenso die etwas dünnen Leiterbahnen im Netzteilbereich, die manchmal etwas "nachdunkeln".
Die Erdung ist eindeutig gelöst; es gibt eine Gehäuseerdung, die 0 Volt sind nicht an Erde geführt.

So, das wars fürs erste. Gruß, Dieter



The Secrets of Beta DOS Calls Teil 4

Orig. von H.Broothaers, uebersetzt von B.Lutz,Tel.07272-6868

Im Nachfolgenden werden die direkten Einsprungadressen fuer die verschiedenen DOS Versionen angegeben. Ich moechte euch allerdings bitten, wenn ihr Programme schreibt, IMMER die in einer der vorhergehenden Beta-Disk Folgen beschriebenen Zentral- adressen zu benutzen, damit das ganze auch jeweils bei anderen Usern mit einer von euch verschiedenen DOS Version richtig laeuft. Vielen Dank.

Direkte Call Adressen fuer die versch. ROM Versionen

Function	3.0	4.03 4.04	4.06	4.07 4.09	4.10	4.1	4.11 4.12
DOS ROM EIN	3CAAh	15467	15467	15467	15467	15467	15467
DISK LESEN	3C09h	11958	11992	11958	11991	11934	11990
DISK SCHREIB.	3COCh	11974	12008	11974	12007	11950	12027
DOS ROM AUS	3CBEh	15484	15484	15484	15484	15484	15484

Vor dem Lese- bzw. Schreib-CALL die Register wie folgt laden:

```
DE =     D =Track,  E = Sector
HL =     Buffer addresse
BC =     B=Anzahl der Sectoren, C=0
```

ACHTUNG: Nicht vergessen das ROM vorher ein und spaeter wieder auszuschalten !!!

Manche werden das Ganze fuer ein bischen langsam halten (was ich nicht verstehe, Haha), also sollte folgende Routine benutzt werden um SCHNELLES Schreiben und Lesen zu erreichen:

Dieses Beispiel ist fuer Version 3.0

```
CALL 3CAAh
ld   HL,Bufferaddresse
ld   D,Start track
ld   E,Start sector
ld   a,Anzahl der Sectoren (was unterschiedlich zu oben ist).
CALL LOADER ROUTINE
hier kann es dann mit dem eigenen Programm weitergehen...
```

LOADER ROUTINE:

```
LD   B,1  (Immer nur ein Sector)
ld   C,0  (C is immer 0 (wie oben))
PUSH AF
PUSH BC
PUSH DE
PUSH HL               (stand nicht im Originalbericht)
Call 3C09h (Muss 3COCh wenn man Schreiben will !!)
POP  HL               (stand nicht im Originalbericht)
INC  H                (stand nicht im Originalbericht)
```

```
  POP  DE
  POP  BC
  INC  E
  LD   A,16
  CP   E
  CALL Z,u(pdate)
  POP  AF
  dec  a
  jr   z,e(nd)
  jr   LOADER ROUTINE (wieder von vorne beginnen)
e call 3CBEh (Das ist der Schluss)
  ret
u LD   e,0
  inc  d
  ret
```

```
 Die folgende Gruppe adressiert den Floppy Disk Controller Chip:
^""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""^
^ PORT OUT                              IN                      ^
^ 1F COMMAND to command reg.            STATUS from status reg. ^
^ 3F TRACK to track reg.                TRACK from track reg.   ^
^ 5F SECTOR to sector reg.              SECTOR from sector reg. ^
^ 7F DATA to data register              DATA from data register ^
^_______________________________________________________________^
```

Folgendes steuert den BETA interface PORT:

FC OUT generiert eine Uhr um einige Schaltkreise im BETA Interf. zu steuern.

```
F7 \
    >-diese beiden  waehrend  OUT  generieren  einen  Takt  zum  Inter  face  um
FF /  Informationen wie Drive Nummer/Disk Seite/ Master Reset  fuer  den  Floppy
      Disk Controller
     -waehrend IN wird ueber sie geprueft ob ein DRQ (Data Request)  oder  INTRQ
      (Interrupt Request) vom FDC anliegt.
      FF wird auch noch benutzt um das Beta ROM ein- bzw. aus zuschalten.
```

Hier sind noch einige der Systemvariablen des DOS:

```
5CC8     Format von drive A
5CC9     format von drive B
5CCA     format von drive C
5CCB     format von drive D
5CCC     track von dem zu Lesen ist
5CCD     sector von dem zu lesen ist
5CCE     00=Lesen // FF=Schreiben
5CD9/A   Effektive LADE Addresse
5CDB/C   Effective Laenge
5CCD     file name (8 lang)
5CE5     file type B/C/D/#
5CE6/7   LADE Addresse fuer code file/Total Laenge fuer Basic
5CE8/9   Pgm. Laenge fuer Basic/Laenge fuer code file
5CEA     file Laenge in Sectoren
5CEB/C   Sector+Track Start des Files
5CF6     Drive Nummer
5CF8     Drive von dem gelesen weden soll
5CF9     Drive auf das geschrieben wird
5D02     Kurzzeitiger Speicher f. Adresse von zu druckendem Text
5D19     Drive Nummer A=0;B=1;C=2;D=3
```

Alle angegebenen Zahlen sind Hexadezimalzahlen !!

Bernhard Lutz (bei Wünschel), Obermühlstr. 24, 6729 Bellheim, Tel. (07272) 6868

Screen-Manipulationen

Heute folgenden weitere Screen-Manipulations-Routinen, von denen ich hoffe, daß mir jemand diese in MC umschreiben kann.

Attr-Kopieren

```
LET a=22528 BIS 22544: LET b=56144: FOR y=1 TO 24: FOR x=1 TO 16: POKE b, PEEK a: LET a=a+1: LET b=b+1: NEXT x: LET a=a+16: NEXT y
```

Attr-Vergrößern

```
LET f=22528: LET d=56144: FOR l=1 TO 384: LET c=PEEK d: POKE f,c: POKE f+1,c: LET f=f+z: LET d=d+1: NEXT l
```

Spiegelung links/rechts

```
LET x=50000: LET z=16383: FOR w=1 TO 192: FOR y=32 TO 1 STEP -1: POKE x,PEEK (z+y): LET x=x+1: Next y: LET z=z+32: NEXT w
LET x=50000: LET z=16384: FOR w=1 TO 6144: LET y=PEEK x: LET t=128: LET s=0: LET r=1: FOR u=1 TO 8: IF y>=t THEN LET y=y-t: LET s=s+r
LET t=t/2: LET r=r*2: NEXT u: POKE z,s: LET z=z+1: LET x=x+1: NEXT w
```

Spiegelung oben/unten

```
FOR x=1 TO 6144: POKE 50000+x,PEEK (16383+x): NEXT x: LET y=56144: LET w=16384: FOR x=1 TO 192: LET y=y-32: FOR v=1 TO 32: POKE w,PEEK (y+v): LET w=w+1: NEXT v: NEXT x
```

Patrick Thiel, Königsberger Str. 11, 4796 Salzkotten, Tel. 05258/5197

Anzeigen

Verkaufe Spectrum 48k Komplett-System eingebaut in PC-Gehäuse, mit abgesetzter PC Tastatur, Beta Disk 4.12 (m. Vision), 3,5" Laufwerk DSDD, 100 Disketten, Originalsoftware, neuer Grundig Datarecorder, Super Rank Xerox Netzteil, Kempston-E-Druckerinterface, Busverlängerung mit 2-fach Verteiler, Umschalter für Fernseher/Videoausgang, Phillips Grünmonitor, Doppelport-Joystickinterface (Kempston+Sinclair I) für 800,- DM.
Weiterhin noch ein Ram Turbo Joystick Interface (Sinclair I+II, Cursor, Kempston, ROM-Karten Anschlußmöglichkeit, durchgeführter Bus) nur 30,- DM. Softwareprogrammierbares DK'tronics Joystick-Interface, durchgeführter Bus: 25,- DM.
Bernhard Lutz (bei Wünschel), Obermühlstr. 24, 6729 Bellheim, Tel. 07272/6868

Verkaufe Spectrum Bücherkiste: ZX Spectrum Tips und Tricks (DATA Becker), ZX Spectrum Hardware Handbuch, ZX Spectrum Hardware (Happy Computer), Handbuch (deutsch), Handbuch (englisch), 3 Spielebücher (je 160 Seiten) komplett für 50,- DM plus Porto.
Für unsere ZX 81 Freaks: Alles über den ZX 81 (Bernstein, 380 Seiten, Programme und Bauanleitungen) für 20,- DM plus Porto.
Für Alphacom Printer: 10 Rollen Thermopapier (originalverpackt) 110 mm. Als Zugabe gibt es einen TIMEX Printer (Stecker defekt). Für 50,- DM plus Porto.
Erhard Gutheim, Bussardweg 1, 3107 Hambühren, Tel. 05084/4553

Verkaufe für den ZX-Spectrum folgende Bücher: Spaß und Profit für den Spectrum; Das Spectrum Buch; 60 Programme für den Spectrum; Alles über Sinclair Computer; Der Sinclair ZX-Spectrum; Einfache Zusatzgeräte für den Spectrum; alle Bücher zusammen für 30,- DM (unbenutzt!)
Außerdem: Jede Menge Kassettenzeitschriften aus England für je 5,- DM; das Programm "Fiendish Freddy's" (128 KB) für 20,- DM und massig Originale 48KB-Spiele.
Suche SAM-Coupe immer noch !!!
Dieter Lederer, Mühlhofstr. 26, 8605 Hallstadt, Tel. 0951/71813
